



Install Manual for a complete version of PLATINE V3.0

V. Baudin

Updated 24/01/2008 16:17:41



Content Page

1 INTRODUCTION.....	5
1.1 BASIC CONCEPTS.....	5
1.1.1 Collaborative e-learning session.....	5
1.1.2 Asynchronous and synchronous phases.....	5
1.1.3 User roles.....	7
1.1.4 Floor control within a synchronous phase.....	8
1.1.5 Flexibility of the approach.....	8
1.2 COLLABORATION MODEL.....	9
2 COLLABORATIVE PLATFORM ARCHITECTURE.....	10
3 GENERIC PLATFORM TOOLS.....	12
3.1 ADMINISTRATIVE SUPPORT TOOLS.....	12
3.1.1 Off-line session tools.....	12
3.1.2 On-line session tools.....	13
3.2 ASYNCHRONOUS TOOLS.....	13
3.3 SYNCHRONOUS TOOLS.....	14
3.3.1 Informal communication among users.....	14
3.3.2 Document sharing.....	14
3.3.3 Multi-user applications.....	15
4 GENERAL ORGANIZATION OF THE FILE SYSTEM FOR THE WHOLE PLATINE. 16	
4.1 DISTRIBUTION OF THE PLATINE CLIENT DATA.....	16
4.2 SESSIONS' SERVER FRONTEND.....	17
5 DESCRIPTION OF PLATINE FILES	18
5.1 FILES OF THE FRONTEND.....	18
5.2 FILES DEDICATED TO FINAL USERS.....	19
6 RUNNING PLATINE.....	21
6.1 SESSIONS' SERVER.....	21
6.2 PLATINE CLIENTS.....	22
ANNEX A - SERVER SIDE.....	25
ANNEX B - PLATINE REQUIREMENTS.....	30
ANNEX C - MULTICASTING	31
ANNEX D - SIGNING JAR FILES.....	32
ANNEX E - PLATINE LICENCE.....	34

1 Introduction

PLATINE is a collaborative work platform which has been developed in the Laboratory for Analysis and Architecture of Systems (LAAS) over the past 8 years. It allows actors of different domains, to work in a collaborative way despite their geographical environmental location. PLATINE has been used in the domain of E-learning and in particular, in the European project Lab@Future (<http://www.labfuture.net/showcase/>)

The purpose of this document is to present some basic concepts used by the PLATINE platform, and to explain how the whole system must be installed. Others documents have been already written for users (administrators of sessions, and participants for some sessions) All those information are available from the PLATINE web site: <http://www.laas.fr/PLATINE> .

PLATINE V3.0 available under the CeCILL-B licence. (see E annex)

1.1 Basic concepts

The purpose of the collaborative platform is to provide all the required communication and collaboration facilities among students and teachers, within a distributed setting, to satisfy the requirements defined by the pedagogic theories for each experiment. To perform distant access to learning experiments, the platform includes a general purpose collaborative platform linked to an experiment specific platform, using specific protocol defined with the Lab@Future project.

1.1.1 Collaborative e-learning session

A collaborative e-learning session is constituted by a group of persons working collaboratively, handling experiment specific data, performing experiment specific applications and using a set of groupware tools as support of their work. The groupware tools provide the basic communication and collaboration services to the users registered in the session.

The size of a session is assumed to be about *ten* persons. As a consequence, the network infrastructure and the server platforms are to be tuned to support the collaborative scenarios for groups of that size.

The collaborative platform may run several concurrent collaborative sessions, which are completely independent.

A session is created, configured and deleted by an *administrator*. An administrator is a specific user who logs into the system with a particular password (there may be either one single password for the collaborative platform, or a password per type of experiment).

1.1.2 Asynchronous and synchronous phases

Two phases are defined within a session, called respectively an *asynchronous* phase and a *synchronous* phase. When a session is created, it enters an asynchronous phase that is continuously active until the session is deleted by the administrator.

In the asynchronous phase, the registered users are authorized to access independently (i.e. without any synchronization among them) the e-learning content of an experiment (for instance, access to experiment specific Web pages) and to perform independently experiment

specific applications. During the asynchronous phase, only asynchronous communication tools, like E-mail, or external communication tools (like POTS) are available to the users.

The administrator is responsible to start and end manually a synchronous phase.

Once a synchronous phase is running, it is the responsibility of each user to explicitly join/leave this synchronous phase. As a consequence, all the users, who have joined a synchronous phase of a session, are aware of each other, and may use different synchronous collaboration and communication tools to work together and implement for example a pedagogic scenario.

It is important to notice that the asynchronous and synchronous phases are not mutually exclusive and may coexist. That is, when a synchronous phase starts, the asynchronous phase keeps active. In this way, users are always allowed to work asynchronously while taking also part in a synchronous activity. Thus, during the lifetime of a session, there are one asynchronous phase and possibly several successive synchronous phases (at any time there is at most one active synchronous phase).

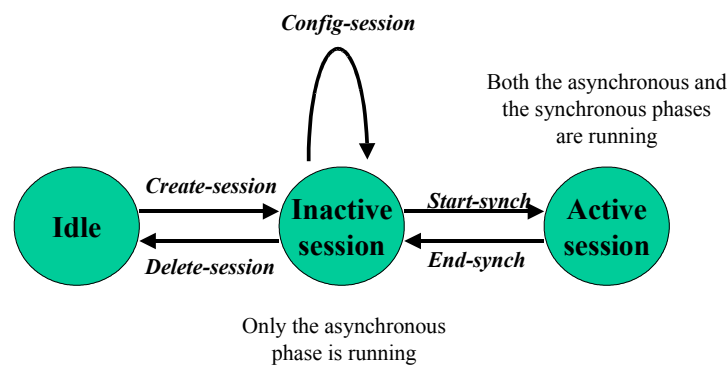


Figure 1. Control states of a collaborative session

We say that a session is *active* if its synchronous phase has been activated by the administrator, otherwise it is said to be *inactive* (i.e. only the tools and data defined for the asynchronous phase are available to the registered users). Figure 1 depicts a state machine representing the control states of a collaborative session, with respect to the interactions of the administrator .

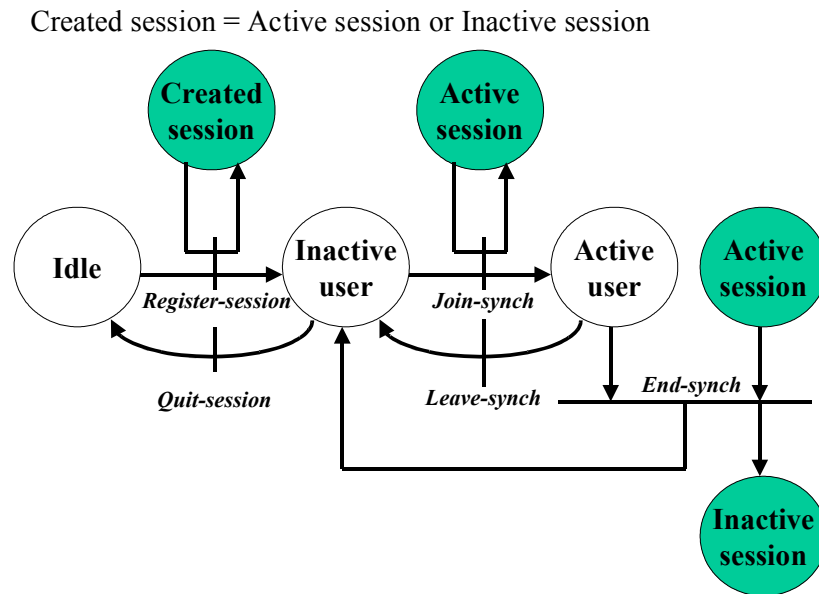


Figure 2. Control states of a user within a session

We say that a user is *active* if he/she joined an active session, otherwise he/she is said to be *inactive*. Figure 2 depicts a Petri net representing the control states of a user, with respect to the interactions of this user and to one interaction of the administrator.

1.1.3 User roles

Besides the *administrator* role which is associated with the user in charge of the management of the session (creation/deletion, configuration of the session), four additional user roles have been defined, which are respectively: *teacher*, *student*, *observer* and *expert*.

Thus, a user, when registering in a session, authenticates him/herself with a specific role. This role determines the rights that the user will have for using the collaboration and communication tools available in the session.

The *teacher* and *student* roles are self-explanatory.

The *expert* role is intended to implement some kind of help desk. It is detained by the users in charge of the problems and questions related to the use of the collaborative platform and/or to experiment specific applications. The help provided by an expert is restricted to the use of the general-purpose collaboration and communication tools and of the experiment specific applications. Other problems related to network configuration and system management are assumed to be out of the scope of the expert and will be supported by standard procedures (i.e. not explicitly supported and controlled by the platform).

The *observer* role is intended to represent a degraded role of the student one. Typically, an observer will have a restricted access to the tools of the collaborative platform. This may be motivated by two main reasons:

- Political reasons related to the session management implying that some users are not allowed to use the full functionality of the platform

- Network QoS constraints implying that some distant users are not capable for technical reasons to use the full functionality of the platform.

The association between user roles and collaboration/communication rights are completely defined by the administrator when configuring the session. They may be modified during the lifetime of a session.

1.1.4 Floor control within a synchronous phase

Another important issue concerns the particular problem of who is in charge of controlling the floor in an active session .

The floor represents a temporary permission to access and manipulate the shared resources. A typical solution to the floor control is the *chaired-based control*, where a human facilitator is in charge of explicitly granting and releasing floor. In the context of this platform, the shared resources are the different collaboration tools that can be executed during the synchronous phase. Then, in this case there is always a central chairman who dynamically grants the floor of each collaboration tool to a different user. For example, suppose that for *tool A*, the chairman grants the floor to *user X*, while for *tool B*, he grants its floor to *user Y*. This means that at this moment, *user X* is the only one allowed to manipulate *tool A* and *user Y* is the only one allowed to manipulate *tool B*.

In order to implement a *chaired-based control*, a *chairman* status has been defined, which may be assigned to one or a subgroup of roles. Persons that are associated with these roles and are active users are authorized to request the chairman status and to potentially gain it. However, this mechanism cannot guarantee the presence of a chairman during the synchronous phase. For example, persons who are authorized to request the chairman status have not yet registered, are not yet active or, if active, have not yet requested the chairman status.

To ensure the presence of a chairman an automaton has been implemented. When the session is started, this automaton is activated and, through a predefined scheduling, it makes the expected floor assignments for the active collaboration tools. Once an authorized user requests the chairman status, the automaton is deactivated and the respective user starts being the session chairman, until he decides not to do anymore. At this moment the automaton is then reactivated.

If an authorized user wishes to request the chairman status while another one holds it, they might use some sort of “social” protocol to negotiate the status switching.

1.1.5 Flexibility of the approach

User roles are roles and not persons. A person can log either with a role or another depending on his knowledge of the required passwords. The unique restriction is that a person connected to the platform through some workstation may always play one and only one role at a time. This means that a person may log, for instance, as administrator and then as teacher and back as administrator, but he cannot behave at the same time as administrator and teacher. However, nothing avoids the same person to connect as teacher from a workstation and as administrator from another workstation. In this case, the same person will concurrently act as a teacher and as an administrator.

There is no a priori restriction on the number of teachers, students, experts and observers who may be connected at a time, unless explicitly specified at the level of the session configuration. Roles are thus defined once by the administrator when creating a new session, and they will not be modified during the session life time. The association between user roles and collaboration/communication rights as well as the assignment of the chairman status are also defined by the administrator, but, contrary to the roles, they may be re-configured by the administrator between consecutive synchronous phases (see Figure 1). Thus some user playing some specific role (for instance, a student) might log a session synchronous phase and not another one, or vice-versa, depending of the current synchronous phase configuration

The execution of a collaboration scenario corresponds therefore to the execution of a session synchronous phase. Switching from one scenario to another at run-time implies a reconfiguration to be made by the administrator. That means that the previous current synchronous phase has to be terminated, the session be re-configured and a new synchronous phase be started.

1.2 Collaboration model

To represent the way users of a collaborative session may exchange information among them, a formal model based on graphs has been introduced. This model is simple enough to be understood and handled in a natural way. It is also general enough to specify different collaborative schemes. The structure of a collaboration scheme is defined by a directed graph, called the collaboration graph, as shown in Figure 3. Vertices represent the members of the collaborative group, and edges the relations between them. Each user within the group is assumed to own a set of information. An edge from user “U1” to user “U2” means that “U1” owns information, which is transmitted to “U2”. Following this approach, “U1” collaborates with “U2” by sending some information to him. Ownership of information still remains to “U1”.

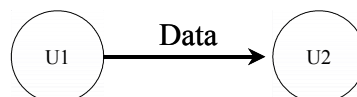


Figure 3.. Graph-based collaboration model

The proposed collaboration model is based on information producer/consumer relationships, to represent and process information exchanges for synchronous collaborative sessions. Such sessions handle continuous data streams (e.g. video, audio), which are specified in a natural way by specific labels associated with the edges of the model.

The application associated with a collaborative group has to define the different group structures that have a meaning for the implementation and the realization of the collaborative work. The collaborative work might be split in separate scenarios, each one being represented by a collaboration graph. Figure 4 presents two examples of scenarios modeled by two collaboration graphs.

The current group configuration evolves from one scenario to another when some specific event occurs within the group.

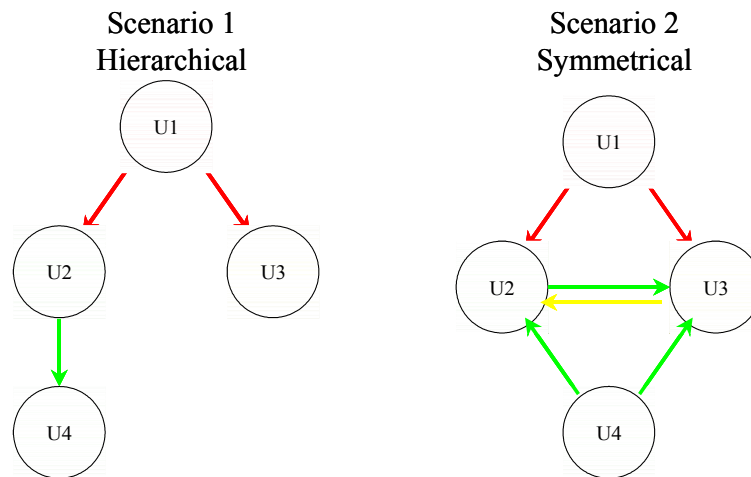


Figure 4. Scenarios and their associated collaboration graphs

To improve the model readability, the graph nodes are colored according to roles associated with them.

The following conventions have been defined:

- Red is associated with the teacher role
- Orange is associated with the expert role
- Green is associated with the student role
- Yellow is associated with the observer role

Arrows from a colored node are currently drawn with the same color as the node, in order to facilitate the identification of the data flow senders.

The collaboration model is used below to characterize the relationships and data exchanges among the members of e-learning sessions.

2 Collaborative platform architecture

The general architecture of the collaborative platform is presented in Figure 5. Main components of this architecture are:

- a set of distributed user workstations interconnected to a WAN network; mobile user workstations, like wireless (802.11 and possibly GPRS) laptops and PDAs, will also be considered;
- a set of distributed servers, four types of servers are considered:
 - The sessions server front-end
 - Experiment specific servers (ESS)
 - Generic collaboration and communication servers (GCCS)

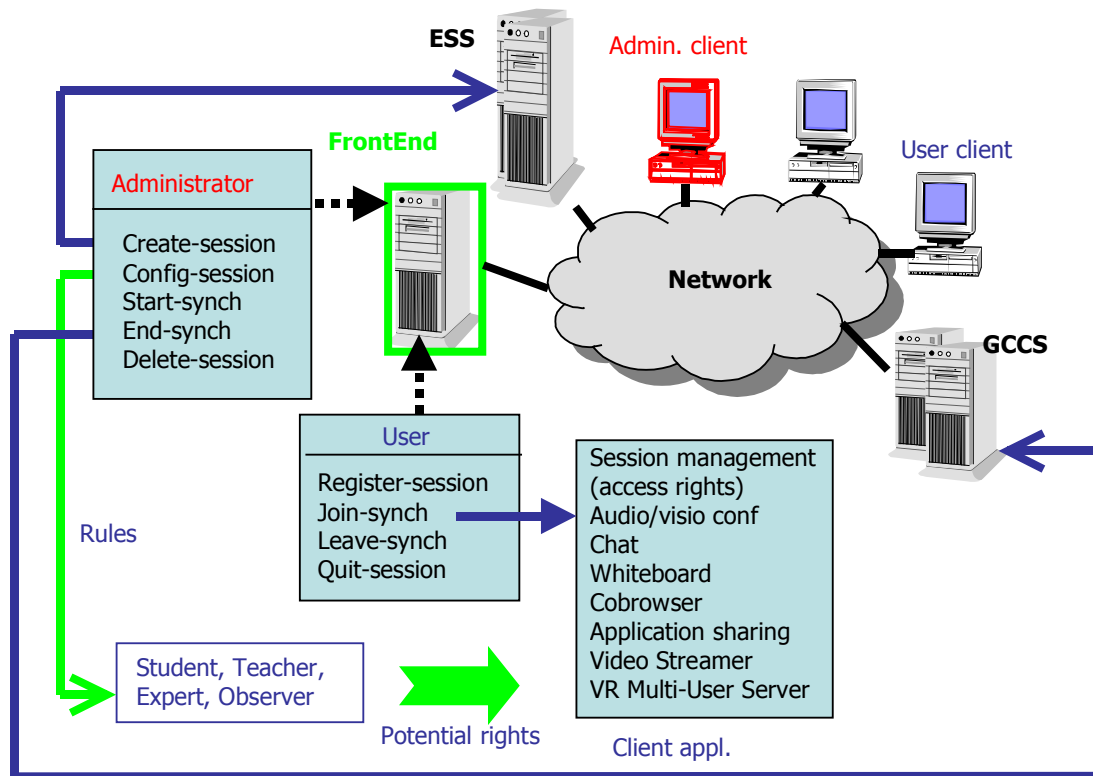


Figure 5. *Architecture of the PLATINE platform*

The set of client workstations for a collaborative session is made of, at most one administrator workstation, and one or more user workstations. The administrator is in charge of creating/deleting and configuring the collaborative session through interactions with the sessions' server front-end. For this purpose, he will use an off-line session preparation tool further detailed in the next section.

Through interactions with the sessions' server, users authenticate themselves to register in a collaborative session with a specific role. Once registered, they automatically take part in the asynchronous phase of the session. They are also responsible to join/leave the session synchronous phase if it is currently active. When registered in a session, users may access experiment specific data (e-learning content) and perhaps perform experiment specific applications. This activity is either completely asynchronous (as part of the session asynchronous phase) or it may be synchronized with the activity of other users in the same session (as part of the synchronous phase). Furthermore, users may have access to synchronous collaboration communication tools during the synchronous phase, like audio-video conferencing, whiteboard, application sharing, etc. These collaboration and communication tools will be detailed in the next section.

The sessions' server is a dedicated front end to access the platform. It provides a common Web interface for supporting the interactions of the administrator and the clients. Users may then access experiment specific data (Web pages, ...), experiment specific applications and collaboration communication tools from a Web browser supporting required plug-ins if necessary. The software tools of the generic collaboration communication servers are automatically activated from the sessions' server when the session becomes active. In a similar way, the software tools of a client workstation are automatically activated from the sessions' server when the respective user becomes active. All these software tools are also automatically deactivated when the session becomes inactive (server and client parts) or when

a client becomes inactive (client part). Furthermore, all the required client software may be downloaded and installed on the client workstation from the sessions' server. These features guarantee the flexibility and transparency of the platform.

Experiment specific servers (ESS) are dedicated data servers storing all the data and applications related to the particular experiment associated with the session.

3 Generic platform tools

The tools implemented in the collaborative platform are classified into three main categories: administrative support tools, asynchronous tools and synchronous tools.

3.1 Administrative support tools

These tools implement the functionalities required for session configuration and management. There are off-line session tools, and on-line session tools.

3.1.1 Off-line session tools

3.1.1.1 Session preparation

This tool provides the administrator with a Web interface for creating and configuring a collaborative e-learning session. The configuration of a session includes the definition of several parameters, including:

- The name of the session
- The user roles considered in the session (a non empty subset of teacher, student, observer and expert)
- The passwords required for registering in the session with a user role
- If needed, the URL of the experiment specific servers accessible to the session users
- The collaboration communication tools available in the session and the definition of their associated IP addresses (IP address of the generic collaboration communication servers, multicast IP address for audio video conferencing, ...)
- If needed, the web pages or a specific application (with web access) that will be automatically opened when the user enters the asynchronous phase and/or the synchronous phase of this session
- The collaboration communication rules associated with the user roles (which user role is authorized to use which collaboration communication tool and how); this feature ensures the flexibility of the platform, since it is possible to adapt for each session the way each user of a specific role is authorized to access collaboration and communication tools
- Which subset of user roles is authorized to get the chairman status for managing the synchronous phase once started.

The session preparation tool will save all the required configuration parameters in an adequate session configuration file.

3.1.1.2 Session registration

This tool provides the users with a Web interface for registering in a session (identification and authentication by session name, user id, password related to the user role) and for quitting a session.

3.1.2 On-line session tools

3.1.2.1 Session synchronous phase management

This tool provides the administrator of a session with a Web interface for managing the start and end of a synchronous phase (in case of the session synchronous phases are managed manually). The tool is in charge of activating/deactivating the server parts of the synchronous collaboration and communication tools.

This tool provides also the users of a session with a Web interface for managing how they can join/leave a session synchronous phase. The module is in charge of activating/deactivating the client parts of the synchronous collaboration and communication tools.

3.1.2.2 Session state display

This tool is in charge of providing user awareness information by displaying the session state components. These components include:

- The name of the session
- The current status of the session (inactive or active)
- The list of the users currently present in the synchronous phase of this session with the indication of their current role (administrator, teacher, student, observer, expert)
- The identification of the user having the chairman status, if any
- The list of all the generic tools configured in the session (these tools are only activated when the session is active)
- For each active generic tool, the identification of the current user to which the floor has been granted; two distinct floors are defined:
 - The document sharing floor required to load or annotate a document with the whiteboard tool, to select and transmit a video with the video streamer
 - The application sharing floor required to interact with a shared application controlled by the application sharing server

3.2 Asynchronous Tools

These tools are used when the users are working in an autonomous way, each one performing his/her own experiment without any synchronous interaction with other users of the session (this corresponds to the session asynchronous phase). If users want to communicate with others, they use asynchronous communication tools like e-mail, or any other communication tool outside of the collaborative platform. Because they are registered in a session, the users are authorized to access experiment specific data (Web pages, VRML

scenes, ...) and applications through classical Web browsers including the required plug-ins .

3.3 Synchronous Tools

Three main collaboration and communication services are available during a session synchronous phase:

- Informal communication among users through audio video-conferencing (in a multicast setting) and chat
- Document sharing through whiteboard and video streaming
- Management of multi-user applications through application sharing or multi-user virtual environments.

3.3.1 Informal communication among users

This service allows users to directly communicate with others through audio, video and text messages while the collaborative activity takes place.

3.3.1.1 Multipoint video/audio conferencing

This tool provides an audio/video informal communication service among the users. It addresses the capture, compression, transmission, decompression and presentation of the multimedia streams and it is based on real or simulated *multicast* communication.

To support real multicast, a native multicast service with an adequate level of QoS, mostly in term of bandwidth requirement, must be available. The videoconference interface is composed of two types of video windows:

- A local video window, captured by the local video camera, monitors the images sent to the users; this window can be hidden to save space on the screen
- The remote video windows display the images from the remote users; they are dynamically updated according to the synchronous group structure modifications.

To ensure compatibility with other current videoconference systems, the video and audio streams are based on the H.263 standard format.

3.3.1.2 Chat

This tool provides a textual informal communication service among users. It can be used in parallel with the video/audio conferencing tool or as a backup informal communication service when the video/audio conferencing tool encounters problems, because it is more robust and less bandwidth consuming.

3.3.2 Document sharing

This service allows users to concurrently look at the same document and possibly to annotate them in a controlled way.

3.3.2.1 Shared whiteboard

This tool provides a graphical document sharing service among the users. It reproduces the behavior of a classical whiteboard in a distributed way. The interface to the shared whiteboard is composed of a graphical window for each user. An authorized user (the one who get the floor) loads images on the whiteboard (for example, JPEG, GIF images) and writes annotations on them. These images and annotations are sent to the other users (the whiteboard clients). These annotations can be saved locally in association with the appropriate images.

3.3.2.2 Video streamer

This tool provides a video document sharing service among the users. It makes it possible to broadcast a previously stored video (audio and video) file in a streaming mode. The tool may also be used to broadcast live video; in this latter case, the camera used by the video streamer tool should not be the same as the one used by the video conferencing tool. (not available for instance)

3.3.3 Multi-user applications

This third service consists of two client-server environments that have in common the ability to allow multiple users to run the same application and to synchronously interact with it.

3.3.3.1 Application sharing

This tool provides users with a generic service for sharing any application. Two different points of view may be considered:

- Any specific application shared during a session is stored on a server; all the users see the execution of the application; an user may interact with the application according to the access right granted by the chairman
- Each user owns an instance of the shared application, with his/her own data; according to the access right managed by the chairman, other users can see and possibly access (in a remote control mode) this application.

In both cases, the client side of the application sharing tool represents a remote view of the shared application. Through the client interface, users can request and, if accepted, accomplish the remote control of the shared application.

The application sharing tool periodically captures and samples the screen dumps of the shared application. They are displayed on a specific video window for each user. It may be then considered as a generic synchronous multi-user tool.

3.3.3.2 Multi-user virtual environments

Some experiment-specific applications provide users with collaborative interaction on the same virtual scene. In order to allow the sharing of interactive virtual scenes, a multi-user virtual environment can be used. This environment is not a part of the PLATINE platform.

4 General organization of the file system for the whole PLATINE

The main goal is to distribute the client part of PLATINE in a secure way, i.e. the right signed version for a specific sessions server.

We have used JavaWebStart to distribute the right version, and to be sure all clients have always the last updated version.

We need a WebServer system: in this document, we install/use APACHE, but it's just an example.

All the needed data for a PLATINE site are separated in two points:

The first one is dedicated to the distribution of the PLATINE client part needed to use this system as a final user or session' administrator

The second one is dedicated to a SuperUser or administrator of the PLATINE site.

4.1 Distribution of the PLATINE client data

On Figure 6, all the needed files are listed.

Some information about the install for APACHE for Windows or for Linux is available on Annex 1.

If APACHE is used as WebServer, the information can be stored at:
/www/htdocs/PLATINE

In this directory, a first one called WebPages contains documents for both administrator of a session, and final user.

Those documents explains how to use PLATINE

for an administrator who wants to create/modify/start/stop/delete a specific session

for a final user who wants to register a session, join or leave the synchronous phase of this session, and quit this session.

We can find also some documents describing some information related to JavaWebStart problems. The list of files on figure6 is just an example.

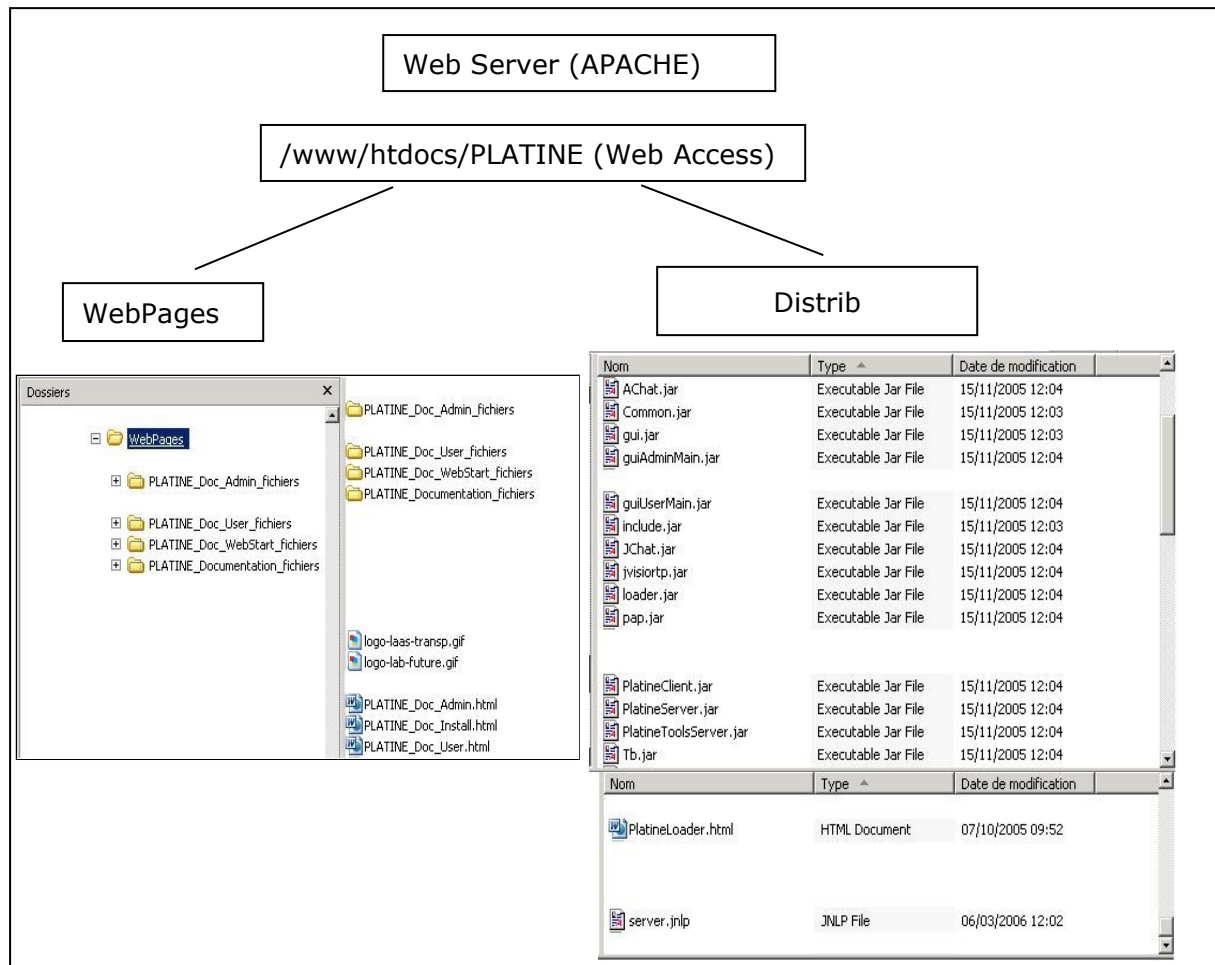


Figure 6. General organization of files downloaded by User

Another directory called `PLATINE_client_files` contains all the jar files needed by a session' administrator or a final user, the jnlp file with the description of the path where all jar files must be stored on the user machine, an HTML file introducing the system for a user with the link to the jnlp file allowing the automated download and start of the application.

Those 2 directories can be stored on a first WebServer machine. A good idea is to use another machine for the FrontEnd of PLATINE: the sessions' server is an active process, and the administrator of this instance of PLATINE has from time to time to modify some data, to restart the sessions' server for any reasons. The administrator of the web site and the administrator of PLATINE can be 2 different persons.

4.2 Sessions' server FrontEnd

Figure 7 depicts the files needed for the PLATINE sessions' server administrator:

- A bat file or a csh file (depends of the OS of his machine) to start the sessions'server
- ThreeTwo data files describing experiments and some technical information.
- Four jar files: PlatineServer, guiSU, AChat, and common

When the sessions' server is started the first time, a new file is created: `GROUND_0.spyConnect`.

This file contains a history of all start and stop of this sessions' server.

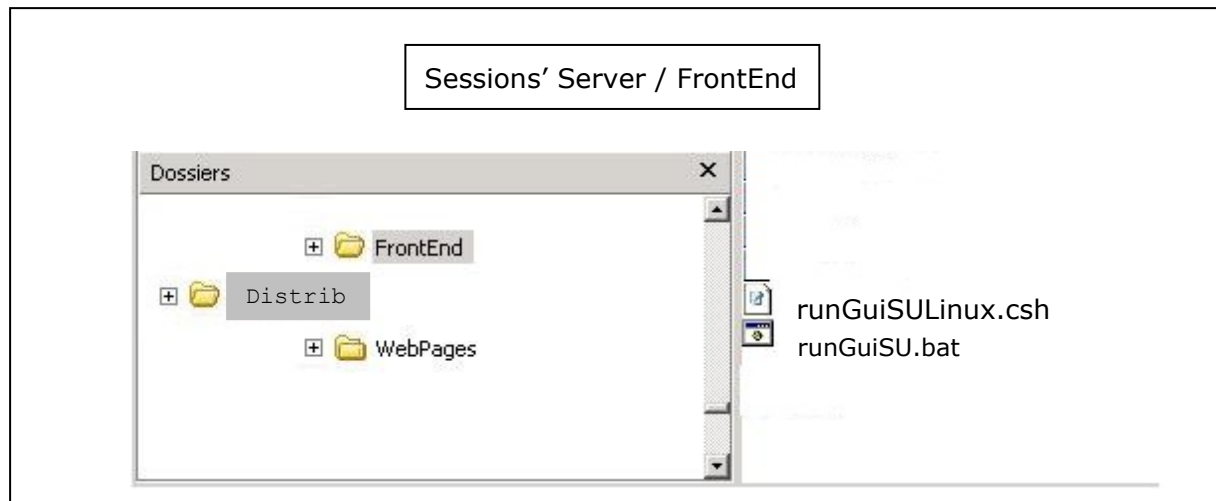


Figure 7. General organization of files for FrontEnd managed by the administrator of the whole system

5 Description of PLATINE files

We describe here the main files of PLATINE for each part Administrator of the whole system and Final User. All the different jar files must be signed. An example is given in Annex D.

5.1 Files of the FrontEnd

The 4 jar files used by the script files of the FrontEnd directory contain all the needed classes to start the sessions' server and to manage all the requests of users (final users and administrator of one session).

runGuiSU.bat and runGuiSULinux.csh contain the command line who start the sessions' server.

Path can be modified to take in account your local install. The first file is dedicated to Windows and the second one to UNIX/Linux.

The next three config files are stored in the distrib directory:

LabFuture contains the password for the general administrator who starts the session's PLATINE server

LabFuture_Config is specific to PLATINE. It describes the experiments managed by the sessions' server. Figure 8 depicts the example used for this distribution.

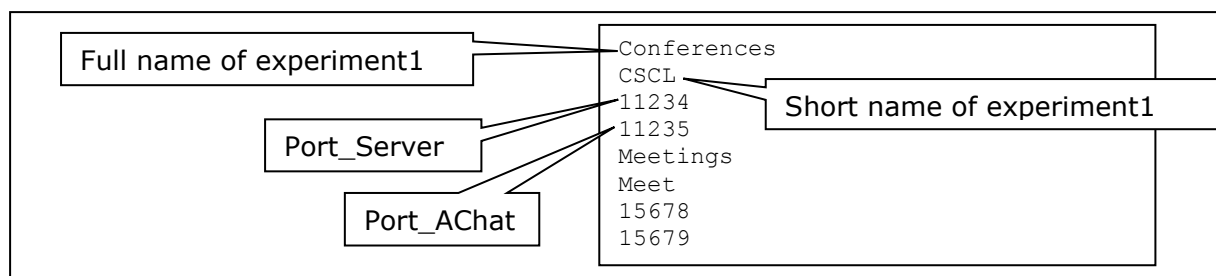


Figure 8. LabFuture_Config file

LabFuture_Config contains 4 information per experiment. The maximum number of experiments is not fixed. Full name and short name for each experiment appears on the interface of sessions' server and the administrator of sessions' server can start some of experiments or just one of them. It appears also on both interface of session's administrator or final user to create/manage a session for the first case, and to register a selected session in one experiment for the second case.

The two following numbers are used for the first one to send and receive information to all users connected to this specific experiment, and for the second one to exchange information through the asynchronous chat between all final users registered on all sessions of this experiment.

Be careful to use different port numbers for all experiments.

LabFuture_Passwords contains the administrator's password for each experiment in the same order of the description of experiments in LabFuture_Config file. The administrator of experiment1 can manage all sessions of this experiment, but he cannot manage sessions of others experiment.

Those two files must be created manually before the first run of sessions' server. To change names of experiment, port numbers, or the list of experiments, a text editor must be used, and the sessions' server must be restarted to take in account the modifications.

When authorized experiments' administrator create sessions, the configuration file for each session is stored in this directory. The name of this configuration file is:

"Experiment_ShortName"."session_name".session.

For the CSCL experiment, a session called cscl05_France is stored in a file called: CSCL.cscl05_France.session

For each experiment a file is also created with the traces of each action' user: it's name is: "Experiment_ShortName".spyConnect.

For the CSCL experiment, this file is called: CSCL.spyConnect

5.2 Files dedicated to final users

Firstly, we can have some userGuide documents. An example is available from the PLATINE web site. Figure 9 depicts the first page of administrator userGuide and final user userGuide.

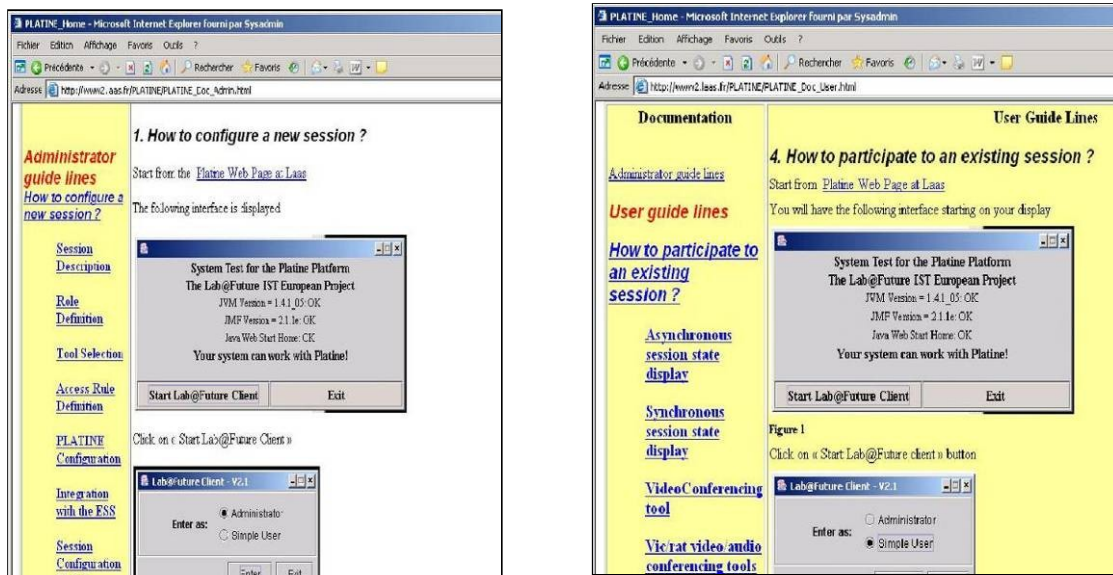


Figure 9. WebPages example

The others files dedicated to final users are stored in PLATINE_client_files. The more important file is "server.jnlp". The following file is given as example: it's the jnlp file for this distribution.

For some explanations about tags, refer to

<http://java.sun.com/products/javawebstart/1.2/docs/developersguide.html>

```

<?xml version="1.0" encoding="utf-8"?>
<!-- JNLP File for WebStats Application -->
<jnlp
  spec="1.0+"
  codebase="http://www.laas.fr/PLATINE/Distrib "
  href="server.jnlp">
    <information>
      <title>PLATINE V3.0 January 21 2008</title>
      <vendor>LAAS - CNRS</vendor>
      <homepage href="http://www.laas.fr/PLATINE/" />
      <description>PLATINE 3.0</description>
      <description kind="short">ST</description>
      <offline-allowed/>
    </information>
    <security>
      <all-permissions/>
    </security>
    <resources>
      <j2se version="1.3+" />
      <jar href="loader.jar" main="true" download="eager"/>
      <jar href="common.jar" download="eager"/>
      <jar href="gui.jar" download="eager"/>
      <jar href="guiSU.jar" download="eager"/>
      <jar href="guiAdminMain.jar" download="eager"/>
      <jar href="guiUserMain.jar" download="eager"/>
      <jar href="PlatineClient.jar" download="eager"/>
      <jar href="PlatineServer.jar" download="eager"/>
      <jar href="PlatineToolsServer.jar" download="eager"/>
      <jar href="include.jar" download="eager"/>
      <jar href="Tb.jar" download="eager"/>
      <jar href="pap.jar" download="eager"/>
      <jar href="jchat.jar" download="eager"/>
      <jar href="AChat.jar" download="eager"/>
      <jar href="jvisiortp.jar" download="eager"/>

    </resources>
    <application-desc main-class="platineguiuser.PlatineLoader">
      <argument>
        .javaws/cache/http/Dwww.laas.fr/P80/DMPLATINE/DMDistrib /
      </argument>
      <argument>1.4.1_05</argument>
      <argument>2.1.1e</argument>
      <argument>pikachu.laas.fr</argument>
    </application-desc>
  </jnlp>

```

Some parameters must be modified, du to your local installation:

Codebase: All relative URLs specified in `href` attributes in the JNLP file is using this URL as a base.

Homepage: Contains a single attribute, `href`, which is a URL locating the home page for the Application. It is used by the Application Manager to point the user to a Web page where they can find more information about the application.

<argument>:

The first argument: the path where all jar files will be stored on the local machine of the final user. In this example www.laas.fr refers to the main web site where PLATINE is installed. This name can be replaced by the IP address of the machine where APACHE server has been installed. We assume that the proposed hierarchy PLATINE/Distrib is used for your local install.

The second and third arguments define the minimum version of Java machine available for this application, and the available version of JMF.

The last argument: the name or the IPaddress where the sessions' server is running. It can be the APACHE server machine, or another one.

A webpage with a link to this jnlp file can be built for the final users. Figure 10 presents such a page.

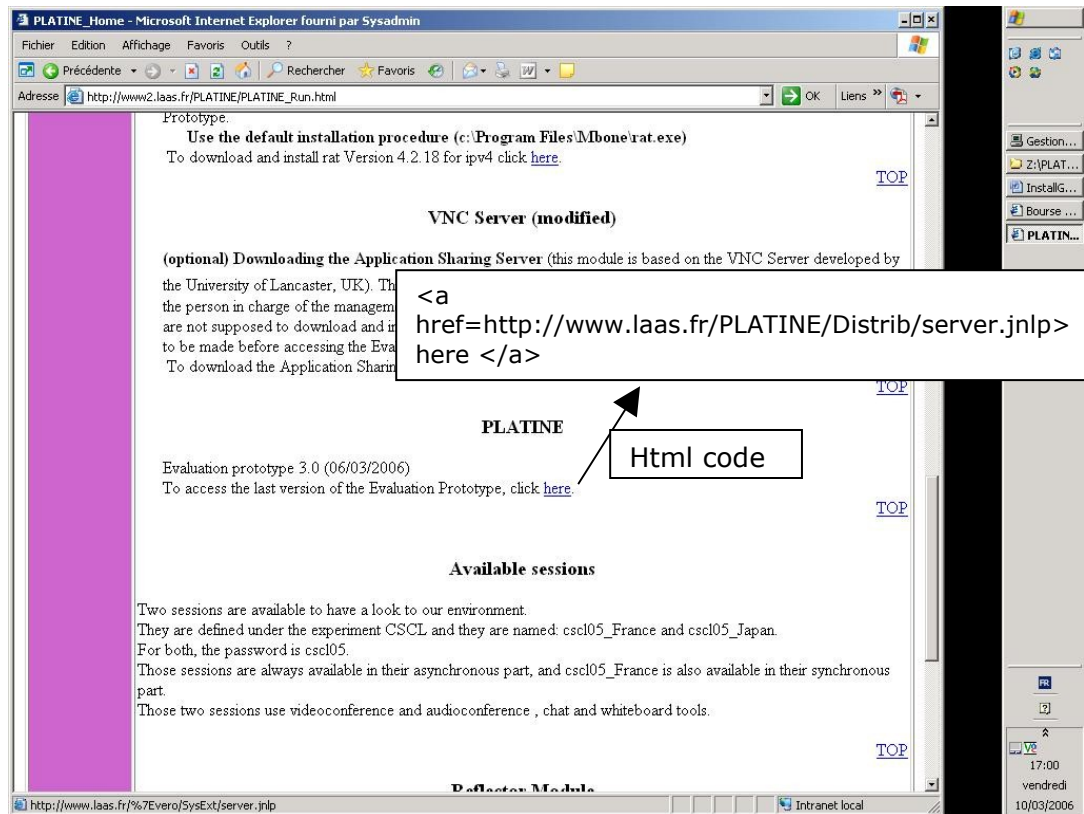


Figure 10. Starting WebPage for PLATINE final user

6 Running PLATINE

6.1 Sessions' Server

For the sessions' server to run, it is important that the java path is correct.

The FrontEnd administrator has to exec the runGuiSU.bat or runGuiSULinux.csh file.

Figure 11 presents the interface of the sessions' server.

The username is su for example, and the password is LAAS. All the experiments defined in the LabFuture_Config file appear on this interface. Click on one checkbox to select an experiment, and on Start button to start the corresponding server for this experiment. This action can be replay for all experiments.

The State button gives information about active experiments, and users connected to the corresponding sessions

The Stop button stops the experiment with the checked checkbox.

The Exit button stops all the started experiments and ends the sessions' server.

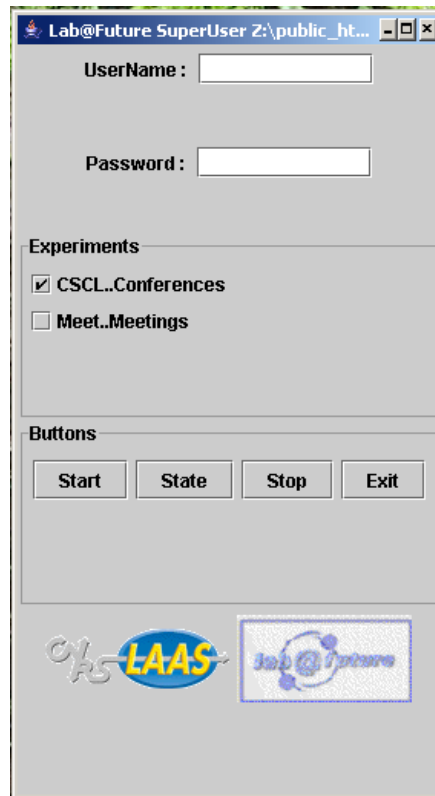


Figure 11. Running Session Server

After entering the username and the password, the experiments available can be selected. As in Figure 11 we have selected CSCL ... Conferences but depending on the super user one or more experiments can be selected and run simultaneously.

6.2 PLATINE Clients

PLATINE clients can access the PLATINE environment via website (the Apache server address where all the necessary documents and links are available to download PLATINE). The clients are advised to install all the necessary Java application on their computers before running PLATINE; this will minimize any unnecessary problems. Some of the screenshot are available below which represent accessing PLATINE from the client side.

Figure 12 presents the result after click on [here](#) (see figure 10)
"server.jnlp" can be started also directly, using the installed JavaWebStart on your test machine.

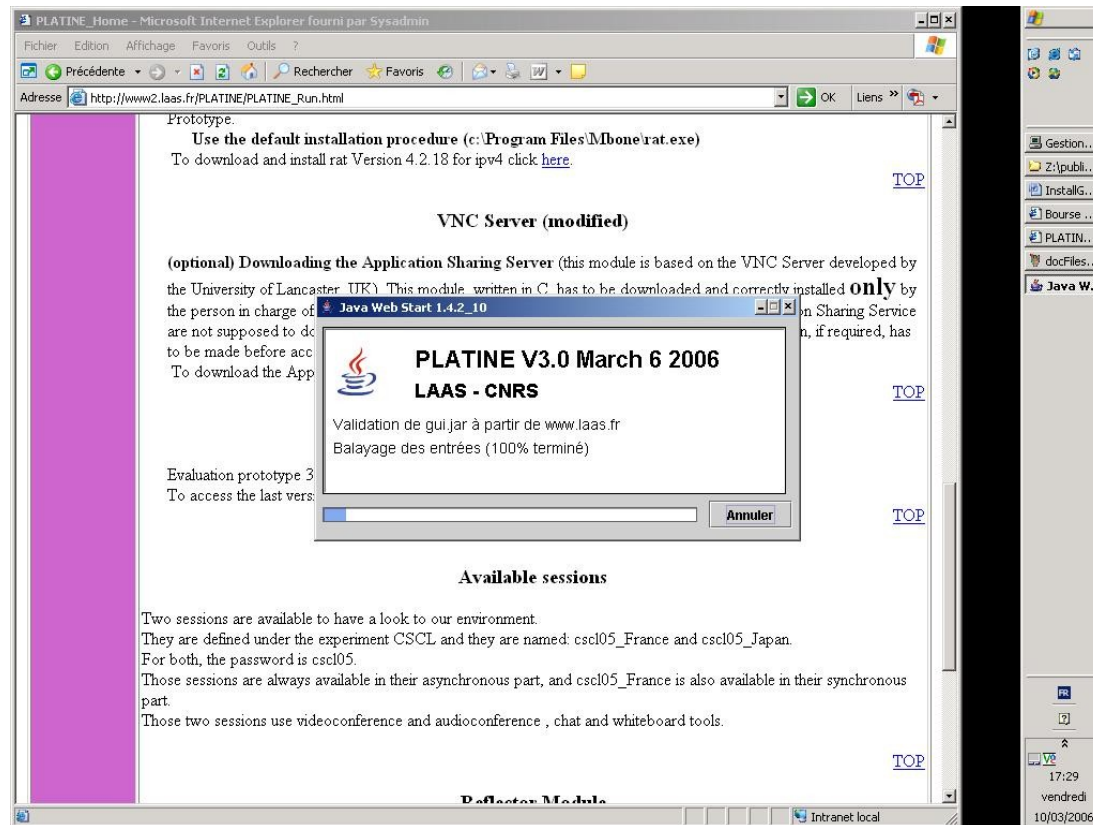


Figure 12. Advertisement window

Figure 12 presents the advertisement before starting PLATINE: in this case a test-certificate is used.

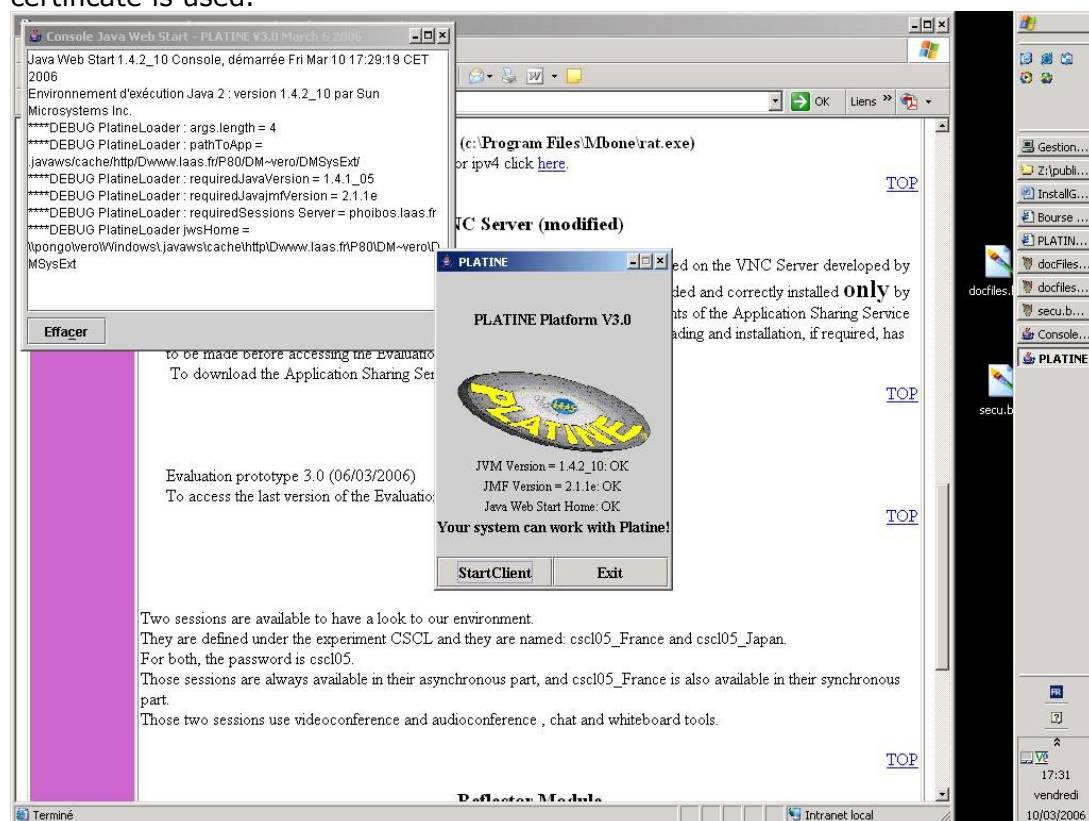


Figure 13. Starting window

Figure 13 presents the first PLATINE interface for session administrator or final user. Those tools are described in the Administrator and FinalUser userguides.

All the sessions created by an authorized administrator are stored in the directory where all jar files are available for download.

Annex A - Server Side

We will be using an Apache Server for our design, which has a reputation of being one of the most powerful web servers. It is often referred to as simply Apache, a public domain open source Web Server. The first version of the Apache server was developed in 1995, based on the National Center for Supercomputing Activities (NCSA) httpd Web server. Since it is an open source, it is freely downloadable and due to its super performance, approximately 58% of web servers around the world are Apache servers.

A.1 Installation of Apache Server for Windows

In this section, we detail how to install and setup an Apache Server for the Microsoft Windows Operating System.

A.1.1 Downloading Apache

It is recommended to download the latest stable version of Apache from [Apache HTTP Server download site](#). Download the binary for Windows 32.

A.1.2 Extracting the Files

Once the binary for Windows 32 is downloaded, run the self installation. - provide step by step screenshots for the procedure.

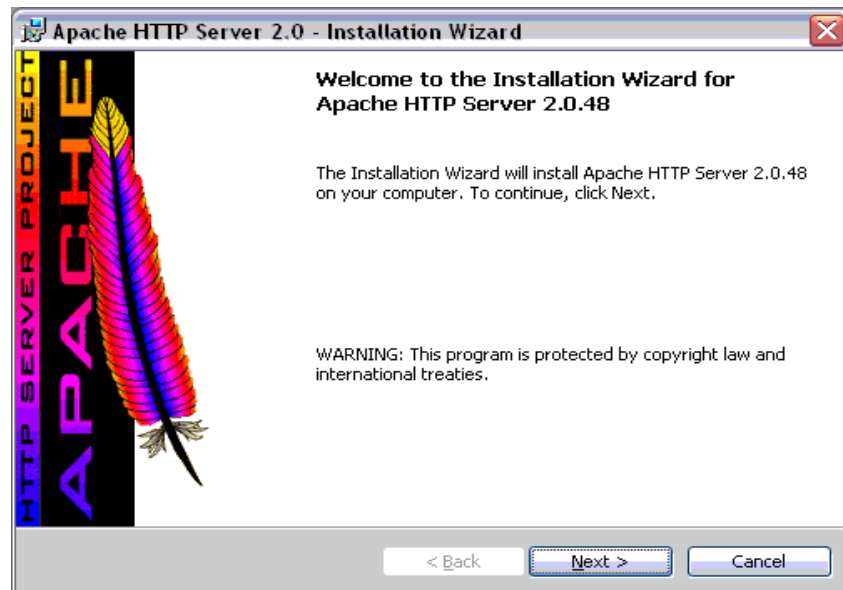


Figure 14. Installation Wizard

This page is the welcome page. Click on next.

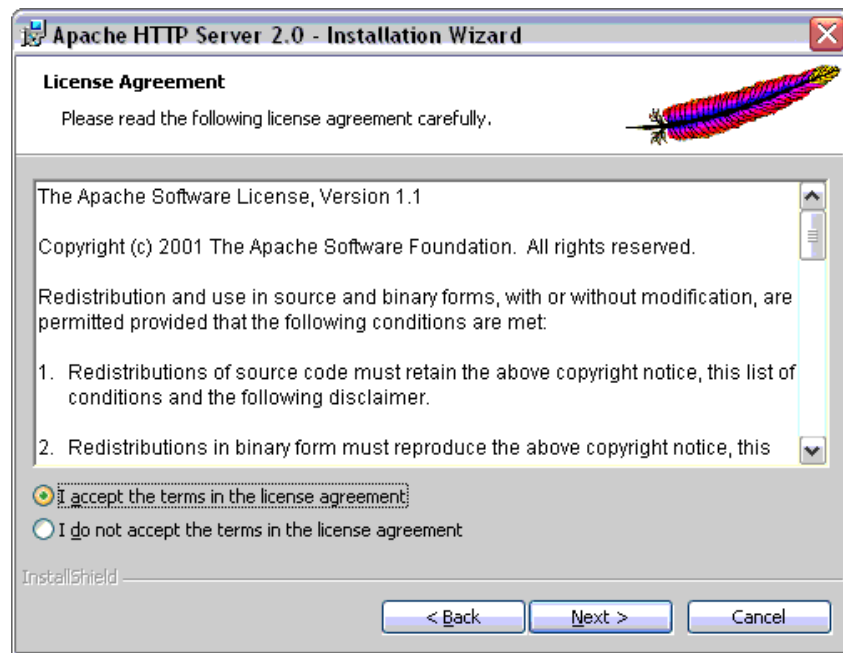


Figure 15. Licence Agreement

This page describes the License Agreement. Accept the agreement and click on next.



Figure 16. Read This First

This README page has lots of helpful information about the installation procedure

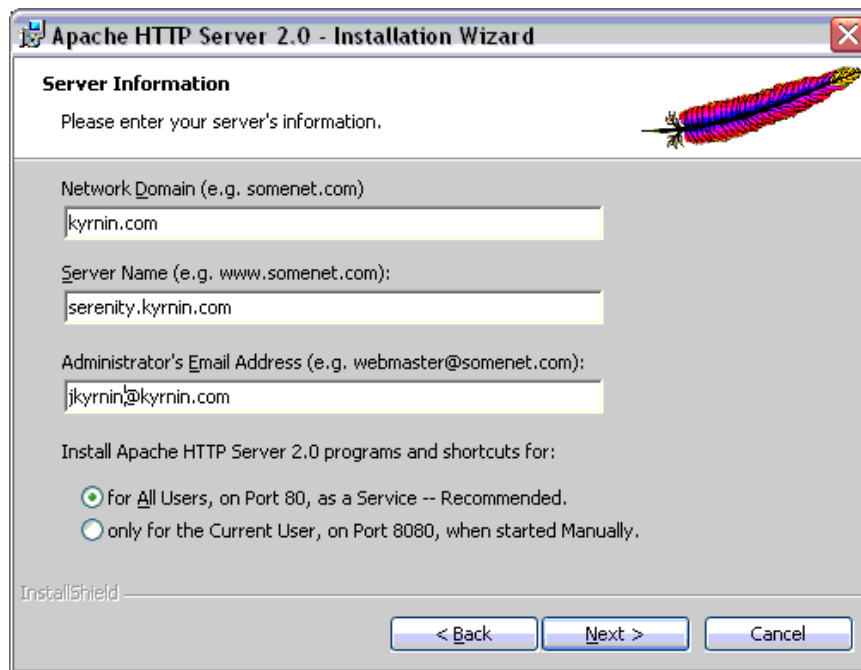


Figure 17. Server Information

Provide your server information here.

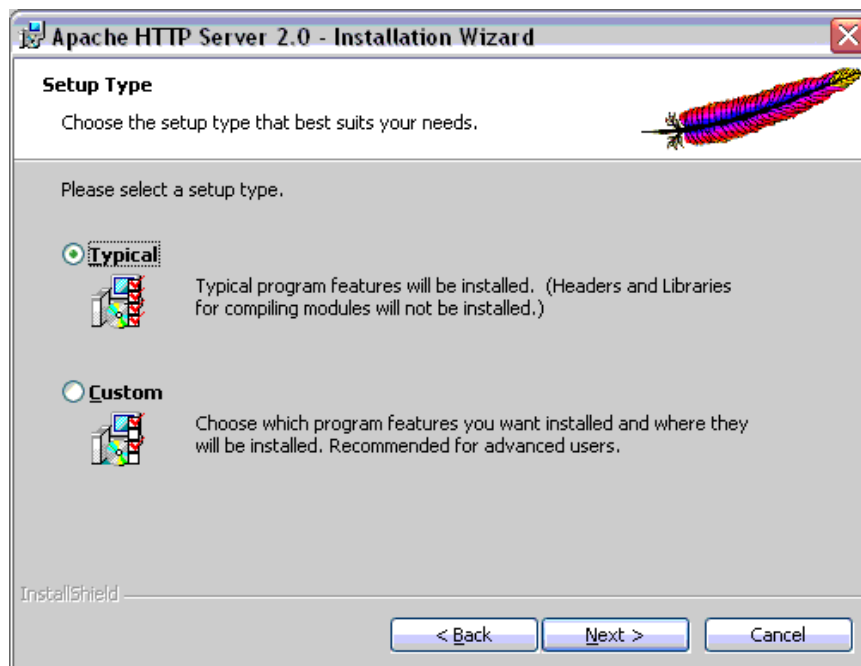


Figure 18. Setup Type

Select a typical or custom setup.

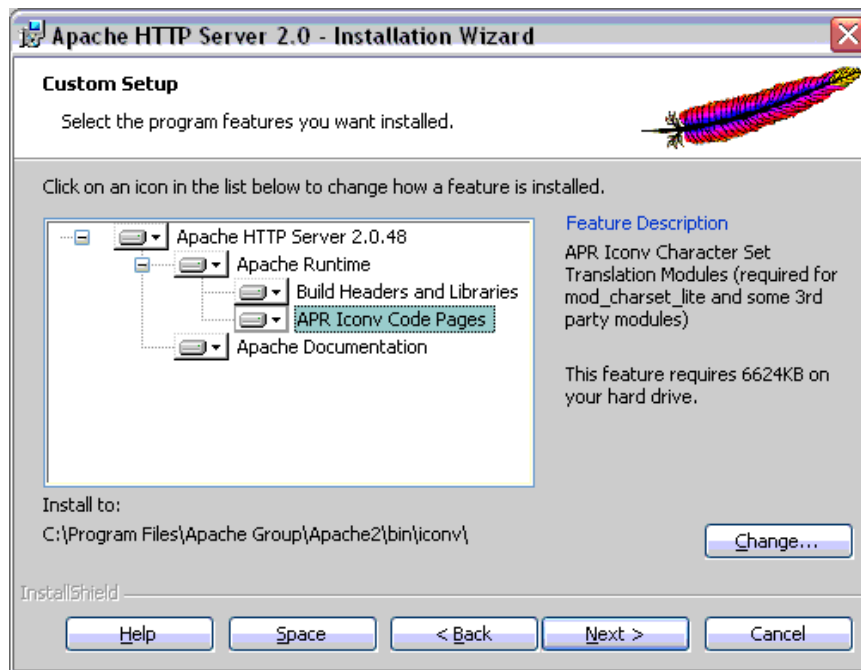


Figure 19. Custom Setup

This screen appears if the custom installation was previously selected.

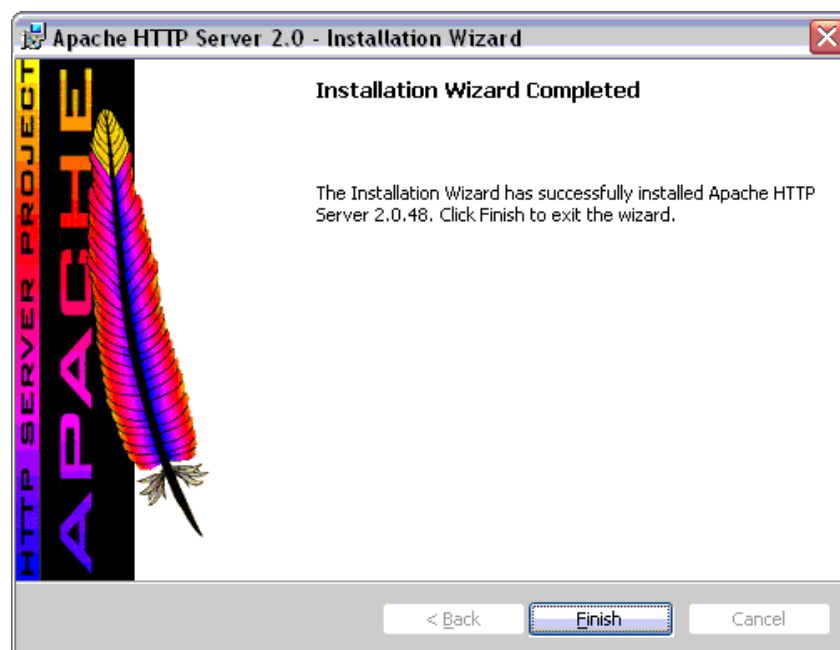


Figure 20. Completed

This final window confirms the installation of the Apache HTTP Server 2.0.

A.1.4 Customisation of the Apache Server

Assuming the above steps are successfully achieved, we can now customise the Apache Server. This is done by editing the `httpd.conf` file which is located in the `C:\Program Files\Apache Group\Apache2\conf` directory. The `httpd.conf` file can be edited by using Notepad or other editing tools. Right-click the `httpd.conf` file and choose 'modify' from the menu to edit the conf file in Notepad.

A.1.5 Testing your Server

Open a web browser on the same machine that you have installed the Apache Server and type <http://localhost/> or <http://127.0.0.1> in the URL address bar. You should see a page saying "Seeing this instead of the website you expected?". This indicates your server is correctly installed.

A.2 Installation of Apache Server for Linux

In this section, we detail how to install and set-up an Apache Server for the Linux Operating System.

A.2.1 Downloading Apache

It is recommended to download the latest stable version of Apache from [Apache HTTP Server download site](#). Download the source appropriate for your system. [Binary releases](#) are available as well.

A.2.2 Extracting the files

Once the files are downloaded uncompress and untar them. This will create a new directory under the current directory with the source file.

A.2.3 Configuring files

Once all the files are received, we need to tell the machine where to find all the files by using `./configure` command to configure the source files.

A.2.4 Build

As with any source installation, the next step is to build the installation by using the commands:

```
make
```

```
make install
```

A.2.5 Customisation of the Apache Server

Assuming that there were no problems in installing Apache, we can now customize the Apache server according to your requirements. Configuring the Apache server is done by making changes to the `http.conf` file by using a text editor such as `vim`, `kwrite` etc.

A.2.6 Testing Your Server

Follow the procedure mentioned in 6.2 to test the Apache server.

A.3 Installing the necessary documents to run PLATINE (All platforms)

Once the Apache server is ready to be used, we have to install all of the necessary documents to start PLATINE on the Apache server. We must allocate a domain name or IP address for the server to make it efficient for all clients to reach the server.

Annex B - PLATINE Requirements

This section highlights the hardware, software and network requirements needed to install PLATINE.

Hardware Requirements	PC with an Intel Pentium 4 or Intel Centrino technology with a minimum of 512Mbytes of RAM memory and 1GHz frequency Good quality audio and video equipment which are recognised by the JMStudio application for end users.
Software Requirements	WebServer like APACHE for example on the website server machine. PLATINE is compatible with: - Windows 2000 - Windows XP - Linux The following Java environment has to be correctly installed on each client machine. JRE or J2SDK Version 1.4.1_05 or higher but not 1.5.0 JMF 2.1.1e
Network Requirements	High speed access to an ISP (Internet Service Provider) at least 512 Kbps providing sufficient amount of QoS. Multicast connectivity can be used for videoconference, but reflector can replace it in some cases (see reflector userguide document on PLATINE web site)

Annex C - Multicasting

PLATINE also supports several users at once therefore providing a collaborative platform for users. Each group is identified by a pair composed of a multicast IPv4 address and a port number. A port number is a positive integer in the range from 0 to 65535. Two levels of multicasting can be distinguished i.e. Local area and Wide area. Local area multicast transmissions take advantage of the possible multicast transmission capabilities of the physical layer by considering Ethernet, which supports point-to-point broadcast and also multicast transmissions. This is made possible by the diffusion nature of the Ethernet technology.

Wide area multicast transmissions uses multicasting routers which are capable of building and managing the multicasting distribution tree. According to whether this multicast router is on a leaf network i.e. with end-hosts connected or on a transit network, it will have different functionality. On a leaf network, the multicast router must discover the presence of local receivers. This is the purpose of the Internet Group Management Protocol v3 (IGMPv3). On a transit network, the multicast router participates in the distribution tree management and multicast packet forwarding.

Annex D – Signing jar files

We use java WebStart for an automatic download on the client machine. To do that, all the jar files must be signed.

An example for signing jar files is proposed below, from:

<http://java.sun.com/j2se/1.4.2/docs/guide/jws/developersguide/development.html#signing>

Signing JAR Files With a Test Certificate

Here are the steps needed to sign a JAR file with a test certificate:

1. Make sure that you have an SDK 1.4.2 `keytool` and `jarsigner` in your path. These tools are located in the SDK bin directory.

2. Create a new key in a new `keystore` as follows:

```
keytool -genkey -keystore myKeystore -alias myself
```

You will get prompted for a information about the new key, such as password, name, etc. This will create the `myKeystore` file on disk.

3. Then create a self-signed test certificate as follows:

```
keytool -selfcert -alias myself -keystore myKeystore
```

This will prompt for the password. Generating the certificate may take a few minutes.

4. Check to make sure that everything is okay. To list the contents of the `keystore`, use this command:

```
keytool -list -keystore myKeystore
```

It should list something like:

```
Keystore type: jks
```

```
Keystore provider: SUN
```

```
Your keystore contains 1 entry:
```

```
myself, Tue Jan 23 19:29:32 PST 2001, keyEntry,  
Certificate fingerprint (MD5):  
C2:E9:BF:F9:D3:DF:4C:8F:3C:5F:22:9E:AF:0B:42:9D
```

5. Finally, sign the JAR file with the test certificate as follows:

```
jarsigner -keystore myKeystore test.jar myself
```

Repeat this step with all of your JAR files.

Note that a self-signed test certificate should only be used for internal testing, since it does not

guarantee the identity of the user and therefore cannot be trusted. A trust-worthy certificate can be obtained from a certificate authority, such as [VeriSign](#) or [Thawte](#), and should be used when the application is put into production.

Annex E – PLATINE licence

PLATINE is distributed under the CeCILL-B licence (<http://www.cecill.info/index.en.html>)
Each PLATINE's file contains the following texte:

Copyright LAAS-CNRS
contributors : V. Baudin, M Diaz, P Owezarski, T Villemur
September, 29 2005

e-mail: veronique.baudin@laas.fr

This software is a computer program whose purpose is to support synchronous cooperative work in a group of users.

This software is governed by the CeCILL-B license under French law and abiding by the rules of distribution of free software. You can use, modify and/ or redistribute the software under the terms of the CeCILL-B license as circulated by CEA, CNRS and INRIA at the following URL "http://www.cecill.info".

As a counterpart to the access to the source code and rights to copy, modify and redistribute granted by the license, users are provided only with a limited warranty and the software's author, the holder of the economic rights, and the successive licensors have only limited liability.

In this respect, the user's attention is drawn to the risks associated with loading, using, modifying and/or developing or reproducing the software by the user in light of its specific status of free software, that may mean that it is complicated to manipulate, and that also therefore means that it is reserved for developers and experienced professionals having in-depth computer knowledge. Users are therefore encouraged to load and test the software's suitability as regards their requirements in conditions enabling the security of their systems and/or data to be ensured and, more generally, to use and operate it in the same conditions as regards security.

The fact that you are presently reading this means that you have had knowledge of the CeCILL-B license and that you accept its terms.